ĐỒ HỌA MÁY TÍNH

(Thực hành với OpenGL- Dành cho SV)

Đặng Văn Đức

dvduc@ioit.ac.vn

HÀ NỘI – 2007/08

MỤC LỤC

Cài đặt OpenGL

I. Visual C/C++ 6.0

Để thực hành được các bài tập trong tài liệu này, ta cần phải cài đặt Visual C/C++ 6.0 và các thư viện OpenGL. Việc cài đặt chúng lên máy tính được thực hiện như hướng dẫn sau đây:

1. Cài đặt *Microsoft Visual Studio 6.0* nếu chưa có nó trong máy tính.
2. Cài đặt OpenGL
   a. Kiểm tra xem *OpenGL v1.1 software runtime* có sẵn trong *WinXP, Windows 2000, Windows 98, Windows 95 (OSR2)* và *Windows NT* hay chưa?
   b. Nếu chưa có, hãy download từ Internet theo địa chỉ: http://download.microsoft.com/download/win95upg/info/1/W95/EN-US/Opengl95.exe
   c. Chạy tệp *Opengl95.exe* vừa tải về để có *OpenGL Libraries* và *header files* sau đây:
      - opengl32.lib
      - glu32.lib
      - gl.h
      - glu.h
3. Cài đặt GLUT
   a. Hãy download từ Internet: http://www.xmission.com/~nate/glut/glut-3.7.6-bin.zip
   b. Cài đặt theo hướng dẫn trong tệp *readme*, và sao chép các tệp như chỉ dẫn sau:
      - glut32.dll vào %WinDir%\System, v*í dụ: C:\windows\system*
      - glut32.lib vào $(MSDevDir)\..\..\VC98\lib, ví dụ: C:\Program Files\Microsoft Visual Studio\VC98\Lib
      - glut.h vào $(MSDevDir)\..\..\VC98\include\GL, ví dụ: C:\Program Files\Microsoft Visual Studio\VC98\Include\gl

Phát triển chương trình ứng dụng bằng Visual C++ và OpenGL

   a. Khởi động *Visual C++* bằng thực đơn *File-New-Projects* để tạo dự án mới "*Win32 Console Application.*"
   b. Chọn "*An empty project*" trong màn hình tiếp theo.
   c. Chuyển đến thực đơn *Project-Settings* trong IDE. Chọn *"Link" tab* và chèn *opengl32.lib, glu32.lib, glut32.lib*, và nếu *project* sử dụng GLUI thì gộp cả *glui32.lib*. Các tệp thư viện này cần có có trong danh mục "*Microsoft Visual Studio\VC98\lib*" hay trong danh mục của *project* hiện hành.
   d. Kiểm tra việc cài đặt bằng cách nhập một chương trình đơn giản GLUT như ví dụ 1.1.
   e. Chú ý rằng ta cần chèn các lệnh #include <GL/glut.h> và #include <GL/gl.h> vào đầu chương trình.
   f. Dịch và chạy thử chương trình.
   g. Sửa lỗi nếu có.
      Nếu xuất hiện thông báo lỗi "*unexpected end of file while looking for precompiled header directive*", hãy tắt thuộc tính "*precompiled headers*" bằng cách chọn *Projects -> Settings*, chuyển đến *C++ tab*, chọn *Precompiled Headers* từ *Category listbox*, sau đó chọn phím radio "*Not using precompiled headers*".

II. Microsoft Visual Studio .NET

1. Thực hiện cài đặt *OpenGL* tương tự như trên đây.
2. Sửa đổi *Project Properties*:
   a. Chọn thực đơn *Project (Project --> * Properties)* của *Visual Studio* để mở trang hộp thoại.
   b. Chọn *combo box 'Configuration'*, chọn '*All Configurat*ion'
   c. Trong pane trái, chọn cây con '*linker'* và chọn '*Input'*. Hãy nhập các dòng sau đây vào trường '*Additional Dependencies'* trong pane phải.
   d. Bây giờ *Visual Studio* biết tìm GLUT ở đâu. Nhấn phím *OK*.

BÀI I. Hệ thống đồ họa

Thí dụ 1.1

Vẽ điểm ảnh trên màn hình.

```
#include <gl/glut.h>
#include <gl/gl.h>

void myDisplay(void) {
	glClearColor(1.,1.,1.,1.);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.,0.,0.);
	//glPointSize(12.0);
	glBegin(GL_POINTS);
		glVertex2i(0,0);
	glEnd();
	glFlush();
}
void main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowPosition(100,150);
	glutInitWindowSize(640,480);
	glutCreateWindow("Thí dụ 1.1");
	glutDisplayFunc(myDisplay);
	glutMainLoop();
}
```

Thí dụ 1.2

Lập trình nhận các phím chuột và bàn phím.

```
#include <gl/glut.h>
#include <gl/gl.h>
#include <stdio.h>

void myDisplay(void) {
	glClear(GL_COLOR_BUFFER_BIT);
	glutSwapBuffers();
	glFlush();
}

void myMouse(int b, int s, int x, int y)
{
	switch (b)
		{		// b indicates the button
		case GLUT_LEFT_BUTTON:
			if (s == GLUT_DOWN)	// button pressed
				printf("\nLeft button pressed!");
			else if (s == GLUT_UP)	// button released
				printf("\nLeft button released!");
			break;
		case GLUT_RIGHT_BUTTON:
			if (s == GLUT_DOWN)	// button pressed
				printf("\nRight button pressed!");
```

```
                else if (s == GLUT_UP)      // button released
                        printf("\nRight button released!");
                break;
                // ...                  // other button events
        default: break;
        }
}

void myKeyboard(unsigned char c, int x, int y)
{
        switch (c)
        {                       // c is the key that is hit
        case 27:                // 'q' means quit
                exit(0);
                break;
        default:
                printf("\nKey %c is hit", c);
                break;
        }
}

void main(int argc, char **argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
        glutInitWindowPosition(300,200);
        glutInitWindowSize(320,320);
        glutCreateWindow("Thi du 1.2");
        glutDisplayFunc(myDisplay);
        glutMouseFunc(myMouse);
        glutKeyboardFunc(myKeyboard);
        glutMainLoop();
}
```

Thí dụ 1.3

Hiển thị xâu ký tự.

```
#include <gl/glut.h>
#include <gl/gl.h>
#include "string.h"

void bitmap_output(int x, int y, int z, char *string, void *font)
{
int len, i;
glRasterPos3f(x, y, 0);         // Locate Raster Position in 3-space
len = (int) strlen(string);       // Find length of string
for (i = 0; i < len; i++) {       // Loop through plotting all characters in font style
    glutBitmapCharacter(font, string[i]);
    }
}

void myDisplay(void) {
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3ub(255, 0, 0);
        bitmap_output(0,0,0, "Hello OpenGL!", GLUT_BITMAP_TIMES_ROMAN_24);
        glFlush();
}
```

```
void main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowPosition(100,100);
	glutInitWindowSize(640,480);
	glutCreateWindow("Thi du 1.3");
	glutDisplayFunc(myDisplay);
	glutMainLoop();
}
```

Thí dụ 1.4

Vẽ điểm ảnh tại vị trí nhấn phím trái chuột.

```
#include <windows.h>
#include <math.h>
#include <gl/glut.h>
#include <gl/gl.h>

void myDisplay(void) {
  glClear(GL_COLOR_BUFFER_BIT);
  glFlush();
}

void myMouse(int button, int state, int x, int y)
{
  int yy;
  yy = glutGet(GLUT_WINDOW_HEIGHT);
  y = yy - y;   /* In Glut, Y coordinate increases from top to bottom */

  glColor3f(1.0, 1.0, 1.0);
  if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN))
		{
		glBegin(GL_POINTS);
		glVertex2i(x, y);
		glEnd();
		}
  glFlush();
}

void myInit(void)
{
	glClearColor(0.0,0.0,0.0,0.0);
	glColor3f(1.0f,1.0f,1.0f);
	glPointSize(2.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
```

```
        glutCreateWindow("Thi du 1.4");
        glutDisplayFunc(myDisplay);
        glutMouseFunc(myMouse);
        myInit();
        glutMainLoop();
}
```

Bài tập 1.1

Vẽ điểm ảnh trên cửa sổ khi di và nhấn phím chuột, với màu C cho trước.

Bài II. Các thuật toán vẽ cơ sở

Thí dụ 2.1

Viết chương trình vẽ đoạn thẳng và hình tròn bằng OpenGL.

```
#include <windows.h>
#include <math.h>
#include <gl/glut.h>
#include <gl/gl.h>

#define PI 3.14159265

class GLintPoint {
public:
	GLint x, y;
};

void drawPoint(GLint x, GLint y) {
	glBegin(GL_POINTS);
		glVertex2i(x, y);
	glEnd();
}

void drawLine(GLint x1, GLint y1,GLint x2, GLint y2 ) {
	glBegin(GL_LINES);
		glVertex2i(x1, y1);
		glVertex2i(x2, y2);
	glEnd();
}

void drawCircle(GLintPoint point, float radius)
{
	float savex, savey;
	const int n = 50;				// number of segments making up arc
	float angle = 0;
	float angleInc = 360.0 * PI / (180 * n);	// angle increment in radians
	savex = point.x;
	savey = point.y;
	for (int k = 0; k <= n; k ++, angle += angleInc) {
		drawLine(savex, savey,
				point.x + radius * cos(angle), point.y + radius * sin(angle));
		savex = point.x + radius * cos(angle);
		savey = point.y + radius * sin(angle);
	}
}

void myDisplay(void) {
	glClearColor(1.,1.,1.,1.);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.,0.,0.);
	drawLine(100, 150, 500, 200);
	glColor3f(0.,0.,1.);
	GLintPoint point;
	point.x = 300;
	point.y = 200;
```

```
	drawCircle(point, 100.0);
	glFlush();
}

void myReshape(int w, int h) {			// window is reshaped
	glViewport (0, 0, w, h);		// update the viewport
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);	// map unit square to viewport
	glMatrixMode(GL_MODELVIEW);
	glutPostRedisplay();			// request redisplay
}

void main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowPosition(100,150);
	glutInitWindowSize(640,480);
	glutCreateWindow("Thi du 2.1");
	glutReshapeFunc(myReshape);
	glutDisplayFunc(myDisplay);
	glutMainLoop();
}
```

Thí dụ 2.2

Vẽ đoạn thẳng có hệ số góc từ 0 đến 1 bằng thuật toán trung điểm.

```
#include <gl/glut.h>
#include <gl/gl.h>

class GLintPoint {
public:
	GLint x, y;
};

void drawPoint(GLint x, GLint y) {
	glBegin(GL_POINTS);
		glVertex2i(x, y);
	glEnd();
}

void midPointLine(int x0, int y0, int x1, int y1)
{
	int dx, dy, incrE, incrNE, d, x, y;
	dx=x1-x0;
	dy=y1-y0;
	d=2*dy-dx;		//dstart
	incrE=2*dy;		//DE
	incrNE=2*(dy-dx);	//DNE
	x=x0; y=y0;
	drawPoint(x,y);
	while(x<x1)
		{
		if(d<=0)	// select E
			{
			d+=incrE;
```

```
                x++;
                }
        else
                {       // select NE
                d+=incrNE;
                x++; y++;
                }
        drawPoint(x,y);
        }
}

void myDisplay(void) {
        glClearColor(1.,1.,1.,1.);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.,0.,0.);
        midPointLine(100, 200, 400, 300);
        glFlush();
}

void myReshape(int w, int h) {                  // window is reshaped
        glViewport (0, 0, w, h);                // update the viewport
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, 640.0, 0.0, 480.0);     // map unit square to viewport
        glMatrixMode(GL_MODELVIEW);
        glutPostRedisplay();                    // request redisplay
}

void main(int argc, char **argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
        glutInitWindowPosition(100,150);
        glutInitWindowSize(640,480);
        glutCreateWindow("Thi du 2.2");
        glutReshapeFunc(myReshape);
        glutDisplayFunc(myDisplay);
        glutMainLoop();
}
```

Thí dụ 2.3

Sử dụng chuột để vẽ đường gấp khúc bằng OpenGL

```
#include <windows.h>
#include <math.h>
#include <gl/glut.h>
#include <gl/gl.h>

const int screenHeight = 480;
const int screenWidth = 640;
int startflag = 1;

class GLintPoint {
public:
        GLint x, y;
};
```

```
void drawLine(GLint x1, GLint y1, GLint x2, GLint y2) {
	glBegin(GL_LINES);
		glVertex2i(x1, y1);
		glVertex2i(x2, y2);
	glEnd();
	glFlush();
}

void myMouse(int button, int state, int x, int y)
{
	static GLintPoint List[2];

	if (startflag == 1) {
		List[0].x = x;
		List[0].y = y;
		startflag = 0;
	}
		if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
		List[1].x = x;
		List[1].y = y;
		drawLine(List[0].x, screenHeight - List[0].y, List[1].x, screenHeight - List[1].y);
		List[0].x = List[1].x;
		List[0].y = List[1].y;
	}
}

void myInit(void)
{
	glClearColor(1.0,1.0,1.0,0.0);
	glColor3f(0.0f,0.0f,0.0f);
	glPointSize(2.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 2.3");
	glutDisplayFunc(myDisplay);
	glutMouseFunc(myMouse);
	myInit();
	glutMainLoop();
}
```

Bài tập 2.1

Vẽ đường tròn bằng thuật toán trung điểm với bán kính và tâm cho trước.

Bài tập 2.2

Viết chương trình vẽ đường gấp khúc bằng thuật toán trung điểmử dụng chuột để chọn tọa độ đầu mút đoạn thẳng.

Bài III. Thuộc tính hình vẽ

Thí dụ 3.1

Vẽ hình với đường viền có thuộc tính (màu, độ rộng và nét đứt)

```
#include <gl/glut.h>
#include <gl/gl.h>

void myInit(void)
{
	glClearColor(1.0,1.0,1.0,0.0);
	glColor3f(0.0f,0.0f,0.0f);
	glPointSize(12.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glEnable(GL_LINE_STIPPLE);
	glLineStipple(4,0x5555);
	glLineWidth(4.);
	glBegin(GL_LINE_LOOP);
		glVertex2i(100,50);
		glVertex2i(100,130);
		glVertex2i(150,130);
	glEnd();
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 3.1");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
}
```

Thí dụ 3.2

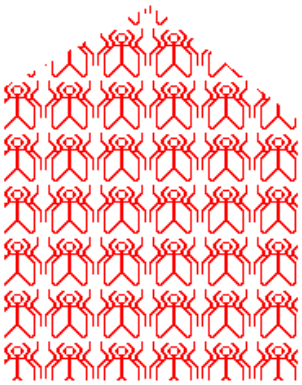
Vẽ đa giác tô theo mẫu.

```
#include <windows.h>
#include <math.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>

void myInit(void)
```

```
{
	glClearColor(1.0,1.0,1.0,0.0);
	glColor3f(0.0f,0.0f,0.0f);
	glPointSize(2.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

void myDisplay(void)
{
	GLubyte mask[]={	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```



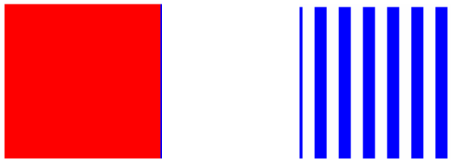
```
			0x03, 0x80, 0x01, 0xC0, 0x06, 0xC0, 0x03, 0x60,
			0x04, 0x60, 0x06, 0x20, 0x04, 0x30, 0x0C, 0x20,
			0x04, 0x18, 0x18, 0x20, 0x04, 0x0C, 0x30, 0x20,
			0x04, 0x06, 0x60, 0x20, 0x44, 0x03, 0xC0, 0x22,
			0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
			0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
			0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
			0x66, 0x01, 0x80, 0x66, 0x33, 0x01, 0x80, 0xCC,
			0x19, 0x81, 0x81, 0x98, 0x0C, 0xC1, 0x83, 0x30,
			0x07, 0xe1, 0x87, 0xe0, 0x03, 0x3f, 0xfc, 0xc0,
			0x03, 0x31, 0x8c, 0xc0, 0x03, 0x33, 0xcc, 0xc0,
			0x06, 0x64, 0x26, 0x60, 0x0c, 0xcc, 0x33, 0x30,
			0x18, 0xcc, 0x33, 0x18, 0x10, 0xc4, 0x23, 0x08,
			0x10, 0x63, 0xC6, 0x08, 0x10, 0x30, 0x0c, 0x08,
			0x10, 0x18, 0x18, 0x08, 0x10, 0x00, 0x00, 0x08};

	glClear(GL_COLOR_BUFFER_BIT);
	glEnable(GL_POLYGON_STIPPLE);
	glPolygonStipple(mask);
	glColor3f(1.0, 0.0, 0.0);
	glBegin(GL_POLYGON);
		glVertex2f(40,40);
		glVertex2f(220,40);
		glVertex2f(220,200);
		glVertex2f(130,270);
		glVertex2f(40,220);
		glVertex2f(40,40);
	glEnd();
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 3.2");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
}
```

Bài tập 3.1

Vẽ hình khép kín với các mẫu tô như sau:



Bài IV. Biến đổi hình học

Thí dụ 4.1

Xoay hình vuông một góc 20 độ. Tâm xoay là gốc tọa độ.

```
#include <gl/glut.h>
#include <gl/gl.h>

void myInit(void) {
	glClearColor(0.7f, 0.7f, 0.7f, 0.0f);		// tô nen màu xám
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);
	glMatrixMode(GL_MODELVIEW);
}

void myDisplay(void) {
	int x=320, y=240;
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0f,0.0f,0.0f);
	glRectf(x-100, y-100, x+100, y+100);
	glColor3f(1.0f,1.0f,0.0f);

	glPushMatrix();				// save the current matrix
	glRotatef(20., 0, 0, 1);		// rotate by 20 degrees CCW in (x,y) plane
	glRectf(x-100, y-100, x+100, y+100);	// draw the rectangle
	glPopMatrix();				// restore the old matrix
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 4.1");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
}
```

Thí dụ 4.2

Xoay hình vuông một góc 20 độ, tâm xoay tại vị trí (x, y).

```
#include <gl/glut.h>
#include <gl/gl.h>

void myInit(void) {
	glClearColor(0.7f, 0.7f, 0.7f, 0.0f);		//Nen màu xám
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, 640.0, 0.0, 480.0);
	glMatrixMode(GL_MODELVIEW);
```

```
}

void myDisplay(void) {
	int x=50, y=40;
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0f,0.0f,0.0f);
	glRectf(200, 100, 400, 300);
	glColor3f(1.0f,1.0f,0.0f);

	glPushMatrix();			// save the current matrix (M)
	glTranslatef(x, y, 0.);		// apply translation (T)
	glRotatef(30., 0., 0., 1.);	// apply rotation (R)
	glTranslatef(-x, -y, 0.);	// apply translation (T)
	glRectf(200., 100., 400., 300.);// draw rectangle at the origin
	glPopMatrix();			// restore the old matrix (M)

	glPointSize(4.);
	glColor3f(0.0f,0.0f,1.0f);
	glBegin(GL_POINTS);
		glVertex2f(x, y);
	glEnd();
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 4.2");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
}
```

Bài tập 4.1

Vẽ hình vuông. Nhấn phím trái chuột, hình vuông xoay xung quanh chúng theo chiều ngýợc kim ðồng hồ. Nhấn phím phải chuột ðể dừng xoay.

Bài tập 4.2

Cho tam giác A(3, 1), B(1, 3), C(3,3).

1. Hãy xác định tọa độ mới của các đỉnh tam giác sau khi xoay một góc 90^0 ngược chiều kim đồng hồ xung quanh điểm P(2, 2).
2. Phóng to tam giác lên hai lần, giữ nguyên vị trí của điểm C. Tính tọa độ các đỉnh tam giác sau khi biến hình.

Bài V. Quan sát ba chiều

Thí dụ 5.1

Hiển thị lập phương từ điểm quan sát cho trước, sử dụng chiếu trực giao.

```
#include <gl/glut.h>
#include <gl/gl.h>

void init(void)
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
}

void myInit(void)
{
	glClearColor(1.0,1.0,1.0,0.0);
	glColor3f(0.0f,0.0f,0.0f);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-3.2, 3.2, -2.4, 2.4, 1, 50);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(8, 10, 7, 0, 0, 0, 1, 0, 0);
}

void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glutWireCube(1);	// Ve lap phuong
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 5.1");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
}
```

Thí dụ 5.2

Hiển thị lập phương từ điểm quan sát cho trước, sử dụng chiếu phối cảnh.

```
#include <gl/glut.h>
#include <gl/gl.h>

void init(void)
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
}
```

```
void myInit(void)
{
	glClearColor(1.0,1.0,1.0,0.0);
	glColor3f(0.0f,0.0f,0.0f);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(45,1,6,100);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(0, 10, 0, 0, 0, 0, 1, 0, 0);
}

void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glPushMatrix();
	glRotatef(45, 1, 1, 1);
	glutWireCube(1);
	glPopMatrix();
	glFlush();
}

void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 5.2");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
}
```

Thí dụ 5.3

Vẽ lập phương trong khối quan sát phối cảnh tổng quát (*Frustum*).

```
#include <GL/glut.h>		// GLUT
#include <GL/glu.h>		// GLU
#include <GL/gl.h>		// OpenGL

void init(void)
{
  glClearColor (1.0, 1.0, 1.0, 0.0);
}

void myDisplay(void)
{
  glClear (GL_COLOR_BUFFER_BIT);
  glColor3f (1.0, 0.0, 0.0);
  glLoadIdentity ();		/* clear the matrix */
        /* viewing transformation  */
  gluLookAt(2.0, 2.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glutWireCube (1.0);
```

```
  glFlush ();
}

void myReshape(int w, int h)
{
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)
{
glutInit(&argc, argv);                    // OpenGL initializations
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400, 400);      // create a 400x400 window
glutInitWindowPosition(0, 0);             // ...in the upper left
glutCreateWindow("Thi du 5.3");      // create the window
glutDisplayFunc(myDisplay);          // setup callbacks
glutReshapeFunc(myReshape);
init();
glutMainLoop();                       // start it running
return 0;                          // ANSI C expects this
}
```

Bài tập 5.1

Vẽ lập phương với từ tập tọa độ cho trước. Điều khiển xoay theo trục x và theo trục y bằng các phím x, y tương ứng. Nhấn phím ESC để thoát.

Bài tập 5.2

Hiển thị lập phương từ điểm quan sát, sử dụng chiếu phối cảnh. Nhấn phím trái chuột để thay đổi góc quan sát fovy.

Bài tập 5.3

Vẽ cánh tay Robot bằng hàm glutWireCube(). Nhấn phím s (*shoulder*) và e (*elbow*) để xoay cánh tay *robot*.

Bài VI. Mô hình hóa ba chiều

Thí dụ 6.1

Vẽ đường cong *Bézier* với 4 điểm điều khiển như sau:
{ -4.0, -4.0, 0.0}, { -2.0, 4.0, 0.0},{2.0, -4.0, 0.0}, {4.0, 4.0,0.0}

```
#include <GL/glut.h>                // GLUT
#include <GL/glu.h>                 // GLU
#include <GL/gl.h>                  // OpenGL
#include <stdlib.h>

GLfloat ctrlpoints[4][3] = {
                { -4.0, -4.0, 0.0}, { -2.0, 4.0, 0.0},
                {2.0, -4.0, 0.0}, {4.0, 4.0, 0.0}};

void init(void)
{
glClearColor(0.0, 0.0, 0.0, 0.0);
glShadeModel(GL_FLAT);
glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &ctrlpoints[0][0]);
glEnable(GL_MAP1_VERTEX_3);
}

void display(void)
{
int i;
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_LINE_STRIP);
for (i = 0; i <= 30; i++)
glEvalCoord1f((GLfloat) i/30.0);
glEnd();
/* The following code displays the control points as dots. */
glPointSize(5.0);
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_POINTS);
for (i = 0; i < 4; i++)
glVertex3fv(&ctrlpoints[i][0]);
glEnd();
glFlush();
}

void reshape(int w, int h)
{
glViewport(0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (w <= h)
        glOrtho(-5.0, 5.0, -5.0*(GLfloat)h/(GLfloat)w, 5.0*(GLfloat)h/(GLfloat)w, -5.0, 5.0);
else
        glOrtho(-5.0*(GLfloat)w/(GLfloat)h, 5.0*(GLfloat)w/(GLfloat)h, -5.0, 5.0, -5.0, 5.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
```

```
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100, 100);
glutCreateWindow ("Thi Du 6_1");
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Thí dụ 6.2

Vẽ mặt cong *Bézier* với tập điều khiển cho trước.

```
#include <windows.h>
#include <stdlib.h>
#include <math.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>

GLfloat ctrlpoints[4][4][3] = {
   {{-1.5, -1.5, 4.0}, {-0.5, -1.5, 2.0},
    {0.5, -1.5, -1.0}, {1.5, -1.5, 2.0}},
   {{-1.5, 0.5, 4.0}, {-0.5, 0.5, 0.0},
    {0.5, 0.5, 3.0}, {1.5, 0.5, 4.0}},
   {{-1.5, -0.5, 1.0}, {-0.5, -0.5, 3.0},
    {0.5, -0.5, 0.0}, {1.5, -0.5, -1.0}},
   {{-1.5, 1.5, -2.0}, {-0.5, 1.5, -2.0},
    {0.5, 1.5, 0.0}, {1.5, 1.5, -1.0}}};

void display(void)
{
  int i, j;

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glColor3f(1.0, 1.0, 1.0);
  glPushMatrix ();
  glRotatef(85.0, 1.0, 1.0, 1.0);
  for (j = 0; j <= 8; j++) {
    glBegin(GL_LINE_STRIP);
          for (i = 0; i <= 30; i++)
                glEvalCoord2f((GLfloat)i/30.0, (GLfloat)j/8.0);
    glEnd();
    glBegin(GL_LINE_STRIP);
         for (i = 0; i <= 30; i++)
                glEvalCoord2f((GLfloat)j/8.0, (GLfloat)i/30.0);
    glEnd();
  }
  glPopMatrix ();
  glFlush();
}
```

```
void init(void)
{
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, 4, 0, 1, 12, 4, &ctrlpoints[0][0][0]);
  glEnable(GL_MAP2_VERTEX_3);
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_FLAT);
}

void reshape(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
    glOrtho(-4.0, 4.0, -4.0*(GLfloat)h/(GLfloat)w, 4.0*(GLfloat)h/(GLfloat)w, -4.0, 4.0);
  else
    glOrtho(-4.0*(GLfloat)w/(GLfloat)h, 4.0*(GLfloat)w/(GLfloat)h, -4.0, 4.0, -4.0, 4.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow ("Thi du 6.2");
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
  return 0;
}
```

Thí dụ 6.3

Cho hai đường cong Bézier P và Q xác định bởi trình tự các điểm sau:

P: A(2, 3, 4), B(3, 1, 5), C(x, y, z), D(3, 4, 3)

Q: D(3, 4, 3), E(2, 6, 0), F(5, 7, 5), G(5, 2, 3)

Hãy thiết lập điều kiện sao cho x, y, z bảo đảm tính liên tục C^1.

Giải:

- P và Q là các đường cong bậc 3:

$$Q(t)=(1-t)^3V_0+3t(1-t)^2V_1+3t^2(1-t)p_2+t^3V_3$$

- Biểu diễn bằng ma trận đường cong Q như sau:

$$Q(t) = \begin{bmatrix}(1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3\end{bmatrix}\begin{bmatrix}V_0\\V_1\\V_2\\V_3\end{bmatrix}$$

- Véc tơ tiếp tuyến (lấy đạo hàm theo t)

$$Q'(t) = \begin{bmatrix} -3(1-t)^2 & 3(3t^2-4t+1) & -3t(3t-2) & 3t^2 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

- Phân đoạn thứ nhất tại cuối đường cong, với u=1, P'(u) và phân đoạn thứ 2 tại đầu đường cong, với t=0, Q'(0)

$$P'(1) = \begin{bmatrix} 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = 3(D-C) \qquad Q'(0) = \begin{bmatrix} -3 & 3 & 0 & 0 \end{bmatrix} \begin{bmatrix} D \\ E \\ F \\ G \end{bmatrix} = 3(E-D)$$

- Để đảm bảo tính liên tục thì P'(u) tại u=1 phải bằng Q'(t) tại t=0

3(D-C)=3(E-D)

3[(3 4 3)-(x y z)] = 3 [(2 6 0)-(3 4 3)]

3[(3-x) (4-y) (3-z)] = 3 [-1 2 -3]

x=3, y=2, z=6

Bài tập 6.1

Vẽ mặt cong Bézier với tập điều khiển cho trước như trong thí dụ 6.2. Bổ sung việc thay đổi tọa độ điều khiển bằng bàn phím.

Bài tập 6.2

Một đường cong Bézier bậc 3 có bốn điểm điều khiển P_0=(2, 2, 0), P_1=(4, 2, 2), P_2=(8, 6, 4), P_3=(12, 0, 0). Giả sử R(t) là đường cong Bézier bậc hai liên tục cấp C^1 tại P_3 của đường cong trên. Đồng thời giả sử R_0, R_1 và R_2 là các điểm điều khiển của đường cong này. Hãy xác định R_0 và R_1.

Bài VII. Loại bỏ mặt khuất

Thí dụ 7.1

Vẽ các khối đa diện bằng OpenGL.

```
#include <gl/glut.h>
#include <gl/gl.h>

void init(void)
{
  glClearColor (1.0, 1.0, 1.0, 0.0);
}

void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0., 0., 1.);

	// Set viewing transformation
	gluLookAt(5., 5., 5., 0, 0, 0, 0., 1.0, 0);

	// Scale cube and display as wire-frame parapellelepiped
	glScalef(1.5, 2.0, 1.0);
	glutWireCube(1.);

	// Scale, translate, and display wire-frame dodecahedron
	glScalef(0.8, 0.5, 0.8);
	glTranslatef(-6., -5., 0.);
	glutWireDodecahedron();

	// Translate and display wire-frame dodecahedron
	glTranslatef(8.6, 8.6, 2.);
	glutWireTetrahedron();

	// Translate and display wire-frame Octahedron
	glTranslatef(-3., -1., 0.);
	glutWireOctahedron();

	// Scale, translate, and display wire-frame icosahedron
	glScalef(0.8, .8, 1.);
	glTranslatef(4.3, -2., 0.5);
	glutWireIcosahedron();

	glFlush();
}

void reshapeFunc(GLint w, GLint h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glFrustum(-1., 1., -1., 1., 2.,20.);
	glMatrixMode(GL_MODELVIEW);
	glClear(GL_COLOR_BUFFER_BIT);
}
```

```
void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640,480);
	glutInitWindowPosition(100,150);
	glutCreateWindow("Thi du 7.1");
	glutDisplayFunc(myDisplay);
	init();
	glutReshapeFunc(reshapeFunc);
	glutMainLoop();
}
```

Thí dụ 7.2

Vẽ và tô màu lập phương với các đỉnh cho trước và bổ sung các phím x, y để xoay lập phương.

```
#include <GL/glut.h>		// GLUT
#include <GL/glu.h>		// GLU
#include <GL/gl.h>		// OpenGL
#include <stdio.h>
#include <math.h>

int a[3]={1.0,1.0,1.0}, b[3]={1.0,-1.0,1.0}, c[3]={-1.0,-1.0,1.0}, d[3]={-1.0,1.0,1.0},
   e[3]={1.0,1.0,-1.0}, f[3]={1.0,-1.0,-1.0}, g[3]={-1.0,-1.0,-1.0}, h[3]={-1.0,1.0,-1.0};

float angle_x=0.0;
float angle_y=0.0;

void drawcube(void) {

 glClear(GL_COLOR_BUFFER_BIT);
 glColor3f(1.0, .0, 0.0);

 glMatrixMode(GL_MODELVIEW);
 glRotatef(angle_y, 0.0, 1.0, 0.0);
 glRotatef(angle_x, 1.0, 0.0, 0.0);
 glBegin(GL_POLYGON);
   glVertex3iv(a);
   glVertex3iv(d);
   glVertex3iv(c);
   glVertex3iv(b);
 glEnd();
 glColor3f(0.0, 1.0, .0);
 glBegin(GL_POLYGON);
   glVertex3iv(a);
   glVertex3iv(b);
   glVertex3iv(f);
   glVertex3iv(e);
 glEnd();
 glColor3f(0.0, 0.0, 1.0);
 glBegin(GL_POLYGON);
   glVertex3iv(d);
   glVertex3iv(h);
   glVertex3iv(g);
   glVertex3iv(c);
 glEnd();
```

```
  glColor3f(1.0, 1.0, 0.0);
  glBegin(GL_POLYGON);
    glVertex3iv(e);
    glVertex3iv(f);
    glVertex3iv(g);
    glVertex3iv(h);
  glEnd();
glColor3f(1.0, .0, 1.0);
  glBegin(GL_POLYGON);
    glVertex3iv(e);
    glVertex3iv(h);
    glVertex3iv(d);
    glVertex3iv(a);
  glEnd();
  glColor3f(0.0, 1.0, 1.0);
  glBegin(GL_POLYGON);
    glVertex3iv(b);
    glVertex3iv(c);
    glVertex3iv(g);
    glVertex3iv(f);
  glEnd();
  glFlush();
}

void myDisplay(void)
{
   glClearColor(1., 1., 1., 1.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (0.0, 1.0, 0.0);
   glLoadIdentity ();             /* clear the matrix */
         /* viewing transformation  */
   gluLookAt(2.0, 3.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
   glEnable(GL_CULL_FACE);
   glCullFace(GL_BACK);
   drawcube();
   glFlush ();
}

void myReshape(int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   glFrustum (-1.0, 1.0, -1.0, 1.0, 1.0, 20.0);
   glMatrixMode (GL_MODELVIEW);
}

void keyboard(unsigned char key, int x, int y)
{
switch (key) {
          case 'x':
          case 'X':
                    angle_x+=10.;
                    glutPostRedisplay();
                    break;
          case 'y':
```

```
        case 'Y':
                angle_y+=10.;
                glutPostRedisplay();
                break;
        case 'i':
        case 'I':
                angle_x=0.;
                angle_y=0.;
                glutPostRedisplay();
                break;
        case 27:
                exit(0);
                break;
        }
}

int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400, 400);       // create a 400x400 window
glutInitWindowPosition(0, 0);               // ...in the upper left
glutCreateWindow("Thi Du 7.2");          // create the window
glutDisplayFunc(myDisplay);               // setup callbacks
glutKeyboardFunc(keyboard);
glutReshapeFunc(myReshape);
glutMainLoop();                         // start it running
return 0;                           // ANSI C expects this
}
```

Thí dụ 7.3

Vẽ và tô màu khối đa diện với các đỉnh cho trước và bổ sung các phím x, y để xoay khối đa diện.

```
#include <GL/glut.h>                // GLUT
#include <GL/glu.h>                 // GLU
#include <GL/gl.h>                  // OpenGL
#include <stdio.h>
#include <math.h>

int a[3]={0.0,0.0,0.0}, b[3]={1.0,-1.0,1.0}, c[3]={-1.0,-1.0,1.0}, d[3]={-1.0,1.0,1.0},
   e[3]={1.0,1.0,-1.0}, f[3]={1.0,-1.0,-1.0}, g[3]={-1.0,-1.0,-1.0}, h[3]={-1.0,1.0,-1.0};

float angle_x=0.0;
float angle_y=0.0;

void drawcube(void) {

  glClear(GL_COLOR_BUFFER_BIT);
  glMatrixMode(GL_MODELVIEW);
  glRotatef(angle_y, 0.0, 1.0, 0.0);
  glRotatef(angle_x, 1.0, 0.0, 0.0);
 // ----------------------------
  glColor3f(1.0, 0.0, 0.0);//red
  glBegin(GL_POLYGON);
    glVertex3iv(d);
    glVertex3iv(h);
```

```
    glVertex3iv(g);
    glVertex3iv(c);
  glEnd();
  glColor3f(0.0, 1.0, 0.0); // green
  glBegin(GL_POLYGON);
    glVertex3iv(e);
    glVertex3iv(f);
    glVertex3iv(g);
    glVertex3iv(h);
  glEnd();
  glColor3f(0.0, 0.0, 1.0);//blue
  glBegin(GL_POLYGON);
    glVertex3iv(b);
    glVertex3iv(c);
    glVertex3iv(g);
    glVertex3iv(f);
  glEnd();
  glColor3f(1.0, .0, 1.0); // magenta
  glBegin(GL_TRIANGLES);
    glVertex3iv(a);
    glVertex3iv(d);
    glVertex3iv(b);
  glEnd();
  glColor3f(0.0, 1.0, 1.0);//cyan
  glBegin(GL_TRIANGLES);
    glVertex3iv(d);
    glVertex3iv(c);
    glVertex3iv(b);
  glEnd();
  glColor3f(1.0, 1.0, 0.0);//yellow
  glBegin(GL_TRIANGLES);
    glVertex3iv(e);
    glVertex3iv(h);
    glVertex3iv(d);
  glEnd();
  glColor3f(0.0, .0, 0.0);//black
  glBegin(GL_TRIANGLES);
    glVertex3iv(e);
    glVertex3iv(d);
    glVertex3iv(a);
  glEnd();
  glColor3f(0.7, 0.7, 0.7);//white
  glBegin(GL_TRIANGLES);
    glVertex3iv(e);
    glVertex3iv(a);
    glVertex3iv(b);
  glEnd();
  glColor3f(.0, .50, .50);//
  glBegin(GL_TRIANGLES);
    glVertex3iv(e);
    glVertex3iv(b);
    glVertex3iv(f);
  glEnd();
  glFlush();
}
void myDisplay(void)
```

```
{
  glClearColor(1., 1., 1., 1.0);
  glClear (GL_COLOR_BUFFER_BIT);

  glClear(GL_DEPTH_BUFFER_BIT);
  glEnable(GL_DEPTH_TEST);

  glLoadIdentity ();             /* clear the matrix */
        /* viewing transformation  */
  gluLookAt(2.0, 3.0, 4.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  drawcube();
  glFlush ();
}

void myReshape(int w, int h)
{
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.0, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

void keyboard(unsigned char key, int x, int y)
{
switch (key) {
          case 'x':
          case 'X':
                    angle_x+=10.;
                    glutPostRedisplay();
                    break;
          case 'y':
          case 'Y':
                    angle_y+=10.;
                    glutPostRedisplay();
                    break;
          case 'i':
          case 'I':
                    angle_x=0.;
                    angle_y=0.;
                    glutPostRedisplay();
                    break;
          case 27:
                    exit(0);
                    break;
          }
}

int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB| GLUT_DEPTH);
glutInitWindowSize(400, 400);              // create a 400x400 window
glutInitWindowPosition(0, 0);              // ...in the upper left
glutCreateWindow("Thi Du 7.3");                    // create the window
glutDisplayFunc(myDisplay);                // setup callbacks
```

```
glutKeyboardFunc(keyboard);
glutReshapeFunc(myReshape);
glutMainLoop();                    // start it running
return 0;                          // ANSI C expects this
}
```

Bài tập 7.1

Vẽ và tô màu hình chóp chữ nhật với các đỉnh như sau:

a={1.0,0.0,0.0}, b={0.0,0.0,-1.0}, c={-1.0,0.0,1.0}, d={0.0,0.0,1.0}, e={0.0,1.0,0.0};

Bổ sung phím x, y và z để xoay hình chóp.

Bài tập 7.2

Vẽ và tô màu hình chóp chữ nhật với các đỉnh như sau:

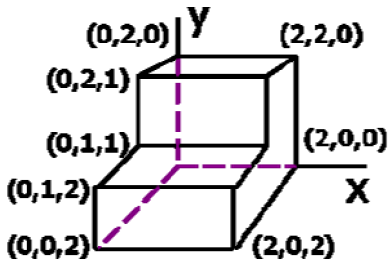
a={2.0,0.0,0.0}, b={0.0,0.0,-2.0}, c={-2.0,0.0,2.0}, d={0.0,0.0,2.0}, e={0.0,2.0,0.0};

Vẽ lập phương với kích thước cạnh là 2.

Bổ sung phím x, y và z để xoay hình chóp.

Bài tập 7.3

Cho đối tượng như hình dưới đây. Hãy xác định các mặt nhìn thấy từ điểm P=(3, 3, -3).



Bài VIII. Chiếu sáng và tô bóng

Thí dụ 8.1

Viết chương trình chiếu sáng hình cầu, sử dụng hàm *glutSolidSphere()* từ một nguồn sáng.

```
#include <GL/glut.h>                // GLUT
#include <GL/glu.h>                 // GLU
#include <GL/gl.h>                  // OpenGL

void init(void)
{
        GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
        GLfloat mat_shininess[] = { 50.0 };
        GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

        glClearColor (1.0, 1.0, 1.0, 0.0);
        glShadeModel (GL_SMOOTH);
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
        glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
        glLightfv(GL_LIGHT0, GL_POSITION, light_position);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_DEPTH_TEST);
}
void display(void)
{
        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glutSolidSphere (1.0, 20, 16);
        glFlush ();
}

void reshape (int w, int h)
{
        glViewport (0, 0, (GLsizei) w, (GLsizei) h);
        glMatrixMode (GL_PROJECTION);
        glLoadIdentity();
        if (w <= h)
           glOrtho(-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w, 1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
        else
           glOrtho (-1.5*(GLfloat)w/(GLfloat)h,1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}

int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize (500, 500);
        glutInitWindowPosition (100, 100);
        glutCreateWindow ("Thi du 8.1");
        init ();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();
```

```
    return 0;
}
```

Bài tập 8.1

Viết chương trình chiếu sáng hình xuyến, sử dụng hàm *glutSolidTorus*(). Bổ sung chuột để thay đổi vị trí chiếu sáng.

Bài tập 8.2

Một mặt phẳng chữ nhật tạo bởi A(0,0), B(1,0), C(1,1) và D(0,1).
Hãy tính cường độ phản chiếu tại điểm P(0.5, 0.5) bằng kỹ thuật tô bóng Gauraud. Biết trước cường độ trung bình của ánh sáng phản chiếu tại bốn đỉnh là: I_A=8, I_B=9, I_C=2, I_D=4.